VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Implementace sledování pohybu objektů pomocí RFID na platformě Raspberry Pi

# Implementation of Object Tracking via RFID on Raspberry Pi Platform

2019 Kristián Rek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Zadání bakalářské práce

Student: **Kristián Rek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Implementace sledování pohybu objektů pomocí RFID na platformě Raspberry Pi
Implementation of Object Tracking via RFID on Raspberry Pi Platform

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je navrhnout a implementovat systém pro sledování objektů pomocí RFID. Konkrétním použitím vzorové implementace bude kontrola chovu hospodářských zvířat. Nicméně systém musí být natolik modulární, aby ho s drobnými úpravami bylo možno nasadit v jiných scénářích se stejnými provozními požadavky. V návrhu bude zvolena vhodná technologie RFID sledování s ohledem na dané použití. Celý systém bude navržen jako distribuovaný. Bude implementována možnost základního kamerového dohledu. V návrhu a v implementaci bude zohledněno ovládání a výstup dat pomocí webové aplikace. Z dalších požadavků plynoucích z implementace pro daný konkrétní scénář bude implementována možnost ovládání dveří a vrat. Funkční řešení bude důkladně otestováno a zdokumentováno.

1. Prostudujte možnosti RFID technologií a platformy Raspberry Pi. Zvolte nejvhodnější technologii pro zadaný scénář použití.
2. Zvolte programovací jazyk a technologie pro implementaci jak samotného systému, tak kontrolní webové aplikace.
3. Proveďte návrh implementace. Dbejte při něm na budoucí rozšiřitelnost a modularitu výsledného řešení a zároveň distribuovanou povahu výsledného řešení.
4. Hotové dílo řádně otestujte a následně zdokumentujte.

Seznam doporučené odborné literatury:

[1] UPTON, Eben; HALFACREE, Gareth. Raspberry Pi user guide. Chichester, West Sussex, UK: Wiley, c2012, 248 p. ISBN 978-1-118-46446-5.
[2] MAS, Roland; HERTZOG, Raphaël. The Debian administrator's handbook: Debian squeeze from discovery to mastery. Lulu Com, 2012. ISBN 979-109-1414-005.
[3] FINKENZELLER, Klaus. Fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication. 3rd ed. Hoboken, NJ: Wiley, c2010, 462 p. ISBN 04-706-9506-4.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Daniel Stříbný**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2019


doc. Ing. Jan Platoš, Ph.D.
*vedoucí katedry*

prof. Ing. Pavel Brandštetter, CSc.
*děkan fakulty*

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2019 .....

Rád bych na tomto místě poděkoval svému vedoucímu Ing. Danieli Stříbnému za cenné rady, připomínky a zapůjčení hardwaru potřebného k vzniku práce.

**Abstrakt**

Cílem práce je seznámit se s technologií stojící za RFID systémy a za pomoci získaných informací vytvořit návrh systému na sledování pohybu objektů a takový systém implementovat.

V teoretické části práce jsou představeny jednotlivé technologie a komponenty potřebné ke sledování objektů pomocí RFID. Na základě funkčních požadavků je pak zvolena vhodná platforma a periferie. Praktická část se zabývá způsobem, jakým jsou komponenty sestaveny, otestovány a implementovány do jednoho systému, který je ovládán pomocí kontrolní webové aplikace. Výsledné řešení je zdokumentováno spolu s návrhem možných vylepšení systému.

**Klíčová slova**: Sledování objektů, RFID, Raspberry Pi

**Abstract**

The goal of this bachelor thesis is to gather information on RFID technology, and use the information in order to create a design for object tracking via RFID and implement it.

In the theoretical part, methods and components needed for object tracking via RFID are introduced, the functional criteria are defined, and a platform and peripherals are selected accordingly. The practical part of the thesis deals with the way the selected components are assembled, tested and implemented in a single system which is controlled via web application. The final solution is documented along with suggestions for potential system enhancements.

**Key Words**: Object tracking, RFID, Raspberry Pi

# Obsah

# Seznam použitých zkratek a symbolů

| | | |
|---|---|---|
| API | – | Application programming interface |
| ARM | – | Advanced RISC Machine |
| CSI | – | Camera Serial Interface |
| CSRF | – | Cross Site Request Forgery |
| DSI | – | Display Serial Interface |
| DTL | – | Django Template Language |
| FFC | – | Flexible Flat Cable |
| GPIO | – | General-Purpose Input/Output |
| GPL | – | General Public License |
| HF | – | High Frequency |
| HTML | – | Hypertext Markup Language |
| $I^2C$ | – | Inter Integrated Circuit |
| IoT | – | Internet of Things |
| IR | – | Infrared |
| LF | – | Low Frequency |
| MVC | – | Model–View–Controller |
| MVT | – | Model–View–Template |
| MQTT | – | Message Queuing Telemetry Transport |
| ORM | – | Object-Relational Mapping |
| PCB | – | Printed Circuit Boards |
| PWM | – | Pulse-Width Modulation |
| RAM | – | Random-Access-Memory |
| RFID | – | Radio-Frequency Identification |
| RS-232 | – | Recommended Standard 232 |
| SBC | – | Single-Board Computer |
| SoC | – | System on a Chip |
| TTL | – | Transistor-Transistor Logic |
| UART | – | Universal Asynchronous Receiver/Transmitter |
| UHF | – | Ultra-High Frequency |
| URL | – | Uniform Resource Locator |
| USB | – | Universal Serial Bus |

# Seznam obrázků

# Seznam tabulek

# Seznam výpisů zdrojového kódu

# 1 Úvod

Kořeny rádiové komunikace sice sahají až do dob druhé světové války, nicméně první patentované RFID (Radio-frequency identification) systémy se začaly objevovat až v letech sedmdesátých. Využití této technologie ovšem stoupá v posledních letech více než kdy jindy. Za tímto růstem stojí hlavně touha po identifikaci a spojení každodenních objektů s Internetem. Tento koncept je nazýván Internet věcí (Internet of things). Mezi populární oblasti aplikace RFID systémů se řadí například obor dopravy, bezpečnosti, bankovnictví a sledování osob či zvířat. A právě řešení problematiky sledování objektů a zvířat je hlavním úkolem této práce.

Nejdříve ale budu řešit z čeho se vlastně takový RFID systém skládá. Prozkoumám typy RFID zařízení a jejich specifické použití v praxi. Po získání dostatečných informací o technologii se pokusím vybrat vhodný čtecí modul pro využití v RFID systému kontroly chovu drobných zvířat. Tento systém bude realizován na platformě kompaktního počítače Raspberry Pi, který je srdcem mnoha podobných projektů. Seznámení se s platformou je obsahem jedné z kapitol.

V další části práce se dostanu k výběru vhodných zařízení pro obohacení funkcionality systému. Mezi další periferie bude patřit motor ovládající vrátka, digitální teploměr a kamera pro zajištění obrazového dohledu. Po otestování jednotlivých periferií se pokusím takovýto systém sestavit a navrhnout optimální způsob napájení a komunikace mezi jednotlivými moduly pro zajištění distribuovanosti systému.

Mým hlavním cílem pro tuto práci je implementovat takový RFID systém, který bude uživatel schopen plně ovládat z webové aplikace. Aplikace musí být optimalizována pro jednoduchou kontrolu pohybu sledovaných objektů či zvířat, ale zároveň bude obsahovat veškeré informace potřebné k reálnému využití.

# 2 Technologie RFID

Radiofrekvenční identifikace (RFID) je rozšířenou formou bezdrátové komunikace využívající elektromagnetické indukce za účelem identifikace a sledování nosiče přiloženého k objektu. Tato technologie patří do skupiny AIDC (Automatic identification and data capture), která se zabývá popisem jednotlivých metod identifikace a sběru informací o objektu. Do této skupiny patří například technologie rozpoznávání čárových kódů BCR (Bar Code Recognition), QR kódů, chytrých karet se zabudovaným mikroprocesorem, OCR (Optical Character Recognition) pro rozpoznávání textu a také řada biometrických metod určených k rozpoznání hlasu, tváře, otisku prstu či sítnice [1].

## 2.1 Komponenty

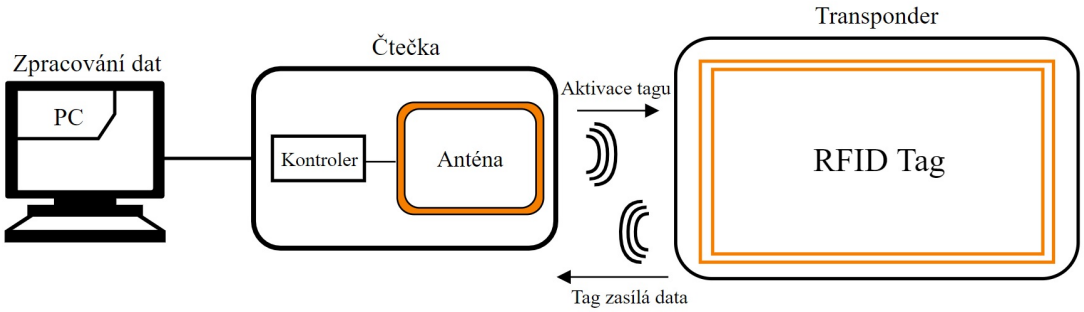
Základem každého RFID systému je tag (transpondér) sloužící jako datový nosič schopný uchovávat informace o objektu. Tag obsahuje čip a integrovaný obvod, který je instalován na izolační vrstvě a slouží jako anténa pro komunikaci s čtečkou. Tagy bývají vyráběny v široké škále velikostí a tvarů umožňujících aplikaci v různých oblastech využití a podle požadavků mohou ukládat i přepisovatelné úložiště dat s kapacitou paměti dosahující až 66 Kbit. Transpondéry jsou nejčastěji vyráběny ve formě klasické PVC karty, klíčového kroužku, disku, připínací/nalepovací pásky, náramku nebo také EPC (Electronic Product Code), keramického či kovového tagu.

### 2.1.1 Transpondér

V závislosti na řešení vlastní napájecí soustavy rozlišujeme dva hlavní typy transpondérů - aktivní a pasivní. Pasivní transpondér nemá k dispozici žádnou formu napájení, pro komunikaci je tedy využito omezeného magnetického či elektromagnetického pole čtečky. Aktivní transpondér má možnost využít energii vlastní baterie k prodloužení operační vzdálenosti komunikace mezi čipem a čtečkou. Aktivní tagy ovšem zpravidla samy o sobě nedosahují přenosového výkonu, který by byl dostatečný ke generování vysokofrekvenčního signálu, a bývají často označovány jako polo-pasivní [3].

### 2.1.2 Čtečka

Za komunikaci, generování signálu a výměnu dat mezi tagem a systémem zpracovávajícím tyto informace je zodpovědná čtečka (viz obr. 1). RFID čtečky můžeme rozdělit na dvě funkcionalitou rozdílné komponenty - radiofrekvenční a řídicí rozhraní. RF (radiofrekvenční) rozhraní se skládá z vysílače a přijímače a má za úkol vyhrazení dostatečného výkonu pro generování přenosového signálu a následnou modulaci tohoto signálu v rámci komunikace s transpondérem. Řídicí komponenta je zpravidla vybavena mikrokontrolérem a modulem pro komunikaci se softwarovou aplikací přes sériové rozhraní.




Obrázek 1: RFID komunikace

## 2.2 Frekvenční pásma

Frekvence, na kterých RFID systémy operují, mají nemalý vliv na vlastnosti a využití takového systému v praxi. Pracovní frekvence má vliv mimo jiné na maximální dosah přenosu, rychlost čtení a zápisu, odolnost proti rádiovému rušení a správnou funkčnost v prostředí se zhoršenými podmínkami (vlhkost, voda, kovy). Před implementací je tedy kriticky důležité provést pečlivý výběr vhodné technologie.

Pasivní a aktivní RFID systémy lze rozdělit podle pracovní frekvence do několika pásem:

- Nízkofrekvenční (LF) RFID - Pokrývají pásmo od 30 kHz do 300 kHz. Z důvodů zahuštění pásma se zpravidla využívají pracovní frekvence 125 kHz a 134 kHz. LF RFID systémy jsou většinou pasivní, mají menší rozsah detekce a disponují lepší odolností vůči okolním rušivým vlivům dřeva, vody a kovu [4]. Oproti ostatním technologiím mají zpravidla také pomalejší rychlost čtení. Na LF RFID tagy obvykle není možno ukládat větší množství dat. Jsou vhodné k využití v systémech řízení a kontrole přístupu do objektu či sledování menší zvěře.

- Vysokofrekvenční (HF) RFID - Operují v pásmu od 3 MHz do 30 MHz s maximální vzdáleností detekce mezi 10 cm a 1 m [2]. Díky pokročilé podpoře šifrovacích protokolů mají široké využití a jsou vhodné pro nasazení v platebních a bezpečnostních systémech. Vysokofrekvenční RFID je definováno jako standard pro sadu NFC (Near Field Communication) technologií.

- Ultra vysokofrekvenční (UHF) RFID - Systémy s rozsahem od 300 MHz do 3 GHz schopné komunikace s transpondérem až na vzdálenosti desítek metrů. Výhodou vyššího pásma je rychlejší datový přenos, ovšem za cenu náchylnosti na vnější rušivé vlivy elektromagnetických vln. Výrobní cena UHF (Ultra-High Frequency) tagů je velice nízká, a proto se využívají v systémech s velkým množstvím sledovaných objektů. UHF je standard pro RAIN RFID - technologii, která se zabývá automatizací identifikace, sběru dat a propojení objektů pro IoT (Internet of Things). Společnost RAIN Aliance byla založena roku 2014 a od té doby dokázala vyrobit a prodat přes 20 miliard čipů [5].

## 2.3 Výběr RFID modulu

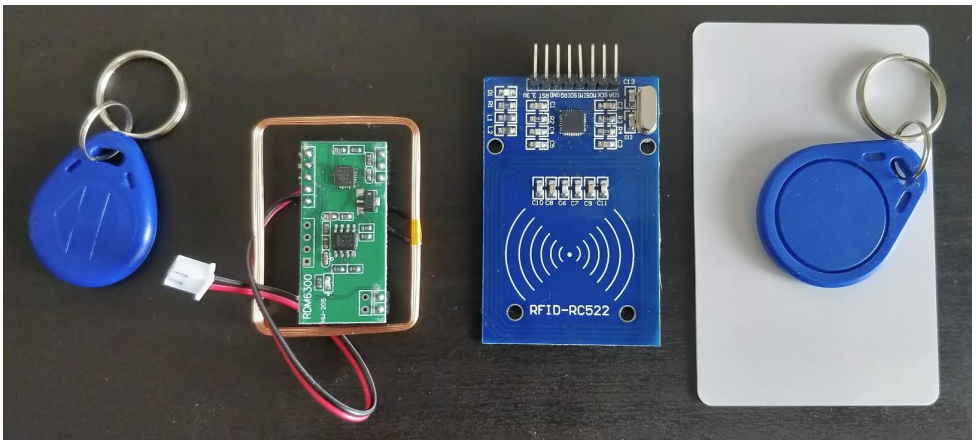
Při výběru optimální technologie k vytvoření vzorové implementace pro kontrolu chovu menších zvířat jsou brány v potaz funkční požadavky takového systému. Pro testování vhodné technologie byly vybrány dva čtecí moduly - RDM6300 a MFRC522 (viz obr. 2). Oba moduly lze pořídit v ceně pod 150 korun za kus a to včetně tagu. Jejich funkce a vlastnosti jsou velmi dobře popsány RFID komunitou, která také stojí za vývojem a údržbou široké škály knihoven pro vývoj na Raspberry Pi, Arduino a jim podobných platformách.

Modul RDM6300 pracuje na frekvenci 125 či 134,2 kHz s maximálním uvedeným dosahem 20 až 50 mm, který je podpořen vnější anténou. Operace je zajištěna 5V stejnosměrným napájením s maximálním odběrem 50 mA a podporovaným komunikačním rozhraním UART (Universal asynchronous receiver-transmitter), TTL (Transistor-transistor logic) a RS-232 (Recommended Standard 232).

Čtečka typu MFRC522 operuje na frekvenci 13,56 MHz s pracovním napětím 3,3 V a uváděnou čtecí vzdáleností dosahující až 50 mm. Podporuje rozhraní I2C (Inter Integrated Circuit) a SPI (Serial Peripheral Interface), obsahuje anténu integrovanou na PCB (Printed Circuit Board) a má výrobcem uveden širší rozsah provozních teplot od -20 do 80 °C [6].

Po zapojení modulů a následném testování s použitím přiložených transpondérů byly zjištěny jisté odchylky od parametrů udávaných v dokumentaci výrobce. U modulu typu MFRC522 se čtecí vzdálenost pohybovala mezi 25 a 30 mm, u modulu RDM6300 pak dosah čtení nepřesáhl 25 mm. Tento výsledek může být ovšem do značné míry ovlivněn kvalitou pasivního obvodu přiloženého tagu. Čtečka MFRC522 má také navrch, pokud jde o rychlost čtení a doby odezvy systému, kde modul RDM6300 přesahoval ve stabilním prostředí dobu 100 ms.

Test obou zařízení potvrdil, že k sestavení systému pro tento scénář je lepší volbou modul MFRC522. Vybraná vysokofrekvenční čtečka může rovněž díky nízkému provoznímu proudu být napájena přímo z platformy Raspberry Pi, aniž by to dramaticky ovlivňovalo funkcionalitu systému.




Obrázek 2: RFID čtečky RDM6300 a MFRC522

# 3 Platforma Raspberry Pi

Raspberry Pi je populární značka SBC (Single-board computer) zařízení, která svými rozměry nepřesahují velikost platební karty. Za svou existenci od roku 2009 vytvořila organizace Raspberry Pi již několik iterací produktů Raspberry Pi a Raspberry Pi Zero. Zařízení tohoto typu našla široké uplatnění v oblasti výuky, integrovaných systémů, robotiky nebo také při každodenním využití jakožto kompaktních počítačů. Varianty modelů se od sebe zpravidla liší počtem rozhraní pro periferie, výkonností SoC (System on a chip) komponent a integrací dalších funkcí pro komunikaci jako jsou Bluetooth a Wi-Fi [8].


Obrázek 3: Raspberry Pi 3 Model B+ [8]

Pro realizaci vzájemné komunikace mezi jednotkami distribuovaného systému jsou v tomto řešení použita zařízení Raspberry Pi 3 Model B a Raspberry Pi 3 Model B+. Systém je díky zpětné kompatibilitě a stejnému seskupení GPIO (General-purpose input/output) pinů možné nasadit na obě zařízení bez jakéhokoliv vlivu na funkcionalitu výsledného řešení.

## 3.1 Hardware

Základem obou modelů je Broadcom čip BCM2837. Ten je vybaven čtyřjádrovým ARM (Advanced RISC Machine) Cortex-A53 clusterem o rychlosti jádra 1.2 GHz a integrovaným multimediálním procesorem VideoCore IV s výchozí frekvencí 400 MHz. Zpracování čipu BCM2837 pro Raspberry Pi 3 Model B+ umožňuje běh na nižších provozních teplotách a je schopen dosáhnout stabilní pracovní frekvence ARM jader přes 1.4 GHz [8]. Zařízení mají kapacitu paměti RAM (Random-Access-Memory) 1 GB, jsou vybaveny 4 USB (Universal Serial Bus) porty, Ethernetovým portem podporující Fast Ethernet (Gigabit Ethernet u modelu B+), PoE (Power over Ethernet) piny a bezdrátovým Wi-Fi standardem 802.11n, který je u modelu B+ rozšířen o 5 GHz spektrum standardem 802.11ac.

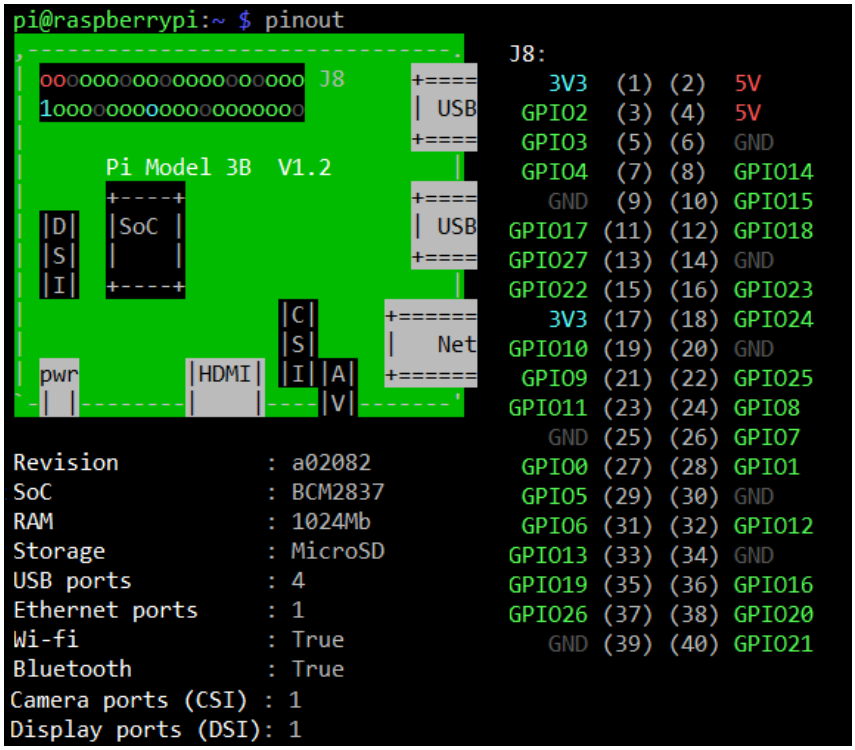
K připojení displeje je dostupný HDMI port a DSI (Display Serial Interface). Raspberry Pi má rovněž integrováno rozhraní CSI (Camera Serial Interface), které bylo využito v tomto systému (viz kapitola 5.3).

## 3.2 Raspbian

Raspbian je vyvíjen a prezentován jako oficiální operační systém pro Raspberry Pi. Jedná se o samostatný projekt, který se specializuje na ARM platformu a je založený na Debianu. Skládá se hlavně z open-source softwaru pod licencí GPL (General Public License). Raspbian lze momentálně stáhnout v několika verzích, já jsem pro účely tohoto projektu použil instalaci Raspbian Stretch s desktopovým prostředím a doporučeným balíkem softwaru. Systém se standardně načítá z SD (Secure Digital), či micro-SD karty o doporučené kapacitě 8 GB. Implementaci této práce lze rovněž uskutečnit s menšími úpravami na operačním systému Ubuntu MATE vyvíjeném pro Raspberry Pi Model B 2, 3 a 3+.

## 3.3 GPIO piny

Přínosem platformy Raspberry Pi 3 je přítomnost 40 GPIO pinů umístěných do dvou řad GPIO portu (viz obr. 3 a 4). K dispozici jsou dva páry pinů pro stejnosměrné napětí 3,3 a 5 V, 8 GND (Ground) pinů pro speciální využití, 26 softwarově řízených GPIO pinů a 2 DNC (Do not connect) piny GPIO00 a 01. K práci s GPIO portem slouží nativní knihovna *RPi.GPIO*.




Obrázek 4: Výstup příkazu pinout s rozdělením GPIO pinů

### 3.3.1 Vlastnosti GPIO

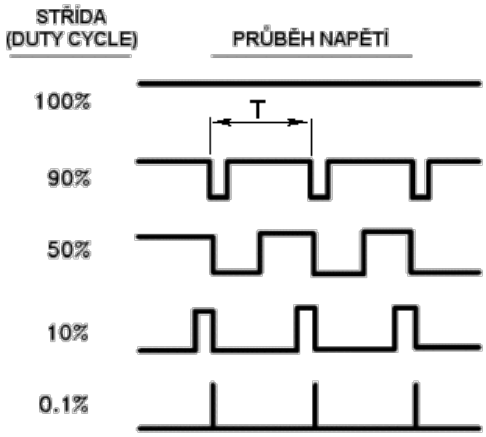
I přes to, že Raspberry Pi má dva piny pro výstup 5 V, pracují GPIO piny na logice 3,3 V a připojení jakéhokoliv zdroje přesahující dané napětí může Raspberry Pi poškodit. GPIO port je vhodný k plnění řady vstupních a výstupních funkcí. Dedikované GPIO piny mohou být uvedeny do jednoho ze tří stavů:

- High - napětí na výstupu je nastaveno na hodnotu 3,3 V.
- Low - výstupní napětí je rovno 0 V.
- Input - nastavení pinu na čtení vstupního napětí.

### 3.3.2 GPIO funkce

GPIO port nabízí i řadu dalších alternativních funkcionalit, které jsou vázány na dedikované piny:

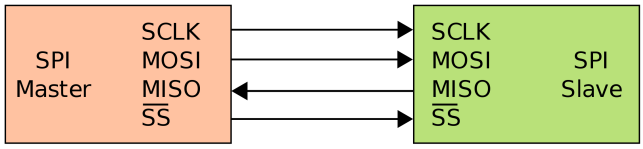
- PWM (Pulse-width modulation) - všechny piny mají možnost využití softwarové pulzně šířkové modulace signálu, která umožňuje efektivní přepínání napětí ve velmi nízkých časových intervalech k regulaci a ovládání širokého spektra zařízení od motorů po žárovky. Softwarové řešení může být instrukčně zatěžující na procesor, proto je k dispozici metoda hardwarového generování a modulace signálu za použití čtyř pinů 12, 13, 18 a 19.




Obrázek 5: Pulzně šířková modulace [10].

- $I^2C$ (Inter Integrated Circuit) - navrženo pro práci s $I^2C$ sběrnicí a zajištění vzájemné komunikace mezi několika integrovanými obvody. U Raspberry Pi slouží k řízení komunikace mezi Broadcom čipem a sérií podřízených (slave) zařízení. Do $I^2C$ sběrnice se přistupuje přes piny 3 a 5. Pin 3 označovaný jako SDA (Serial Data Line) slouží k přenosu dat mezi zařízeními a pin 5 - SCL (Serial Clock Line) - je určen k přenosu řídícího signálu generovaného master obvodem [16].

- SPI (Serial Peripheral Interface) - Rozhraní umožňující, stejně jako $I^2C$, komunikaci mezi integrovanými obvody za pomoci SPI sběrnice (viz obr. 6). U Raspberry Pi je možné




Obrázek 6: Sběrnice SPI s řídicím a podřízeným zařízením

pro tuto komunikaci použít pět pinů a obsluhovat maximálně dvě slave zařízení - pin 19 zajišťuje MOSI (SPI Master Output, Slave Input) signál, pin 21 obsluhuje MISO (Master Input, Slave output) signál, pin 23 je určen ke generování SLCK (Serial Clock) signálu pro synchronizaci komunikace a piny 24 a 26 jsou označovány jako SS (Slave Select), které slouží k určení podřízeného zařízení ke komunikaci [17]. SPI rozhraní je v této práci použito k řízení komunikace s RFID modulem MFRC522 (viz kapitola 5.1).

- UART (Universal Asynchronous Receiver/Transmitter) - rozhraní určeno pro sériovou komunikaci realizovanou na dvou vodičích v asynchronním módu vhodné například ke komunikaci s modemy, RFID čtečkami a širokou škálou modulů. Připojením UART sběrnice k zařízení, které je schopné zobrazit data, můžeme vidět zprávy Linux *kernelu*. To se hodí například k diagnostice při problému s bootováním systému [7]. Pin 8 s označením TXD je určen pro vysílací (transmit) signál a pin 10, označený jako RXD, pro signál přijímaný (receive). Přenos dat UART je definován modulační rychlostí dosahující až 115200 Baudů (nejčastěji 9600 Baudů). Rozhraní používá datové rámce o maximální velikosti 9 bitů a nepodporuje master-slave systémy [9].

# 4 Technologie a periferie systému

V této kapitole se věnuji výběru technologií a modulů pro realizaci testovacího systému kontroly chovu hospodářských zvířat. Při volbě periferií a metod implementace byl kladen důraz na kompatibilitu, možnosti využití nativních nástrojů pro vývoj a testování systému na platformě Raspberry Pi. Rovněž je brán v potaz požadavek na přenositelnost systému s důrazem na oddělení logiky práce s hardwarovými komponenty systému a webové aplikace.

## 4.1 Volba nástrojů pro implementaci

Programovací jazyk zvolený k implementaci funkcionalit systému je Python. Využitím jazyka Python dostáváme k dispozici nepřeberné množství literatury, dokumentace a knihoven vytvořených jak výrobci programovatelných zařízení, tak komunitou uživatelů. Distribuce Raspbian momentálně využívá verze Python 2.7.13 a 3.5.3, které byly vydány na přelomu roku 2016/2017. Abych předešel možným problémům s kompatibilitou starších knihoven, rozhodl jsem se pro verzi 2.7.13. Pro budoucí projekty by kvůli končící podpoře verzí 2.7.x bylo pravděpodobně lepší zvolit variantu druhou.

Malou výjimkou použití budou knihovny $SPI-Py$ a $Motion$. Knihovna $SPI-Py$ je použita ke komunikaci s RFID čtečkou přes rozhraní SPI. Knihovna byla vyvinuta v jazyce C a přepsána jako rozšíření pro Python. Projekt $Motion$ je kamerový sledovací software s řadou funkcionalit, který je také napsaný v jazyce C a umožňuje komunikaci s širokou škálou kamer.

Pro ukládání záznamů o zvířatech, zpráv systému, teplot a řídících instrukcí je nasazena databáze SQLite, která má pro toto množství tabulek adekvátní předpoklady, a díky své jednoduchosti se zaslouží o celkové snížení zatížení systému.

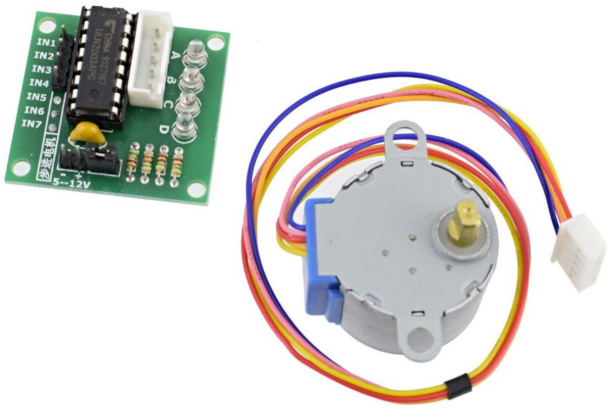
Webová aplikace byla vyvinuta použitím webového frameworku Django. Framework je zcela napsán v jazyce Python, exceluje v oblasti vývoje webových aplikací na slabších strojích a zajištění jejich bezpečnosti. Django následuje architekturu MVC (Model–View–Controller) respektive MVT (Model–View–Template), která umožňuje použití již implementovaných částí systému a jednoduchou rozšiřitelnost o další aplikace. Jednou z největších výhod frameworku Django je možnost použití již existujících, ale plně modulárních, metod řešících autentizaci, formuláře pro registraci, administraci přístupu, správu obsahu a mnoho dalších důležitých kroků vývoje internetových aplikací.

Distribuovanost systému je zajištěna implementací protokolu MQTT (Message Queuing Telemetry Transport) pro zasílání zpráv mezi odesílatelem (publisher) a adresátem (client). MQTT funguje na architektuře klient-server s využitím protokolu TCP/IP. O spojení, zasílání a kontrolu řídících paketů se stará zprostředkovatel (broker). Zprávy jsou definovány pro předem nastavená témat. Klient, který je připojený přes zprostředkovatele a je přihlášený k odběru těchto témat, pak zprávu po publikování obdrží. Výhodou této metody jsou nízké požadavky na rychlost síťového připojení a podpora zpráv dosahujících až 268 MB dat [15].

## 4.2 Motor pro vrátka

K otestování řešení kontroly otevírání a zavírání vrátek byl vybrán krokový motor 28BYJ-48. Společně s motorem je dodáván základní řídicí modul ULN2003 se čtyřmi kontrolními LED indikátory pro testovací účely (viz obr. 7) [11]. Motor pracuje na stejnosměrném napětí 5 V s maximálním provozním proudem řadiče 500 mA. Převodový poměr motoru je 1:64, což nám udává úhel kroku 5,625°. Pro demonstraci distribuovanosti řešení obsahuje systém motory dva - motor na straně klienta je ovládán pomocí MQTT zprávy z hlavního zařízení.

Při výběru motoru v praxi je také potřeba myslet na váhu dveří a možnosti použití dalších snímačů pro přesnou kontrolu polohy dveří. Systémy náchylné na vibrace a hluk motoru mohou profitovat použitím menších kroků (microstepping) za cenu možnosti ztráty točivého momentu [12]. Opakem bude situace, kdy se výkon motoru stane nedostačujícím. V takovém případě je možné zvýšit proud cívky na maximální možnou hodnotu uváděnou výrobcem, čímž dosáhneme vyššího točivého momentu. Řídicí moduly motorů však bývají na změny proudu často náchylné. Další možnou variantou je aplikace vyššího napětí. Na rozdíl od varianty první, tato metoda ovlivní hlavně rychlost otáček motoru. Je ovšem zapotřebí dbát na omezení daná konstrukcí a materiálem cívky motoru.



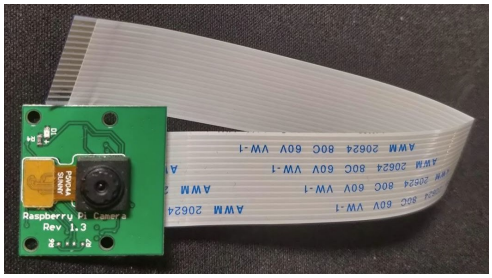

Obrázek 7: Řídicí modul ULN2003 s motorem 28BYJ-48 [11].

V rámci testování jsem rovněž vyzkoušel aplikovat infračervený senzor FC-51 určený pro vyhýbání se překážkám. Senzor mi pomohl garantovat úplné otevření/zavření a detekci vniku nechtěných objektů do cesty vrátek. IR (Infrared) senzor se zkládá z infračerveného vysílače a přijímače. Detekce překážky je zajištěna odrazem infračerveného záření od daného objektu a následnou detekcí záření přijímačem. Snímač při detekci pošle na výstup 3,3 V signál, který je rozpoznám GPIO pinem. Nevýhodou tohoto řešení je požadavek na stabilní světelné podmínky, neboť schopnost odrazu klesá s tmavším objektem a prostředím. Senzitivitu detekce je proto nutné řídit potenciometrem.

## 4.3 Kamerový dohled

Kamerový dohled je realizován za pomoci knihoven projektu *Motion*. *Motion* kombinuje funkce vysílání živého videa, detekce a aktivace při zaznamenání pohybu, ukládání záznamů a publikování na web s možností zabezpečení záznamu. Knihovna je zároveň kompatibilní s většinou existujících *Video4Linux* ovladačů, IP kamer a různých nahrávacích zařízení [14].

Záznam obstarává kamerový modul OmniVision s objektivem OV5647 (viz obr. 8). Modul je zapojen do Raspberry Pi pomocí FFC (Flexible flat cable) a komunikuje přes CSI rozhraní. Kamera je schopna pořizovat snímky v rozlišení 5 Mpx (2592 x 1944) a nahrávat 1080p videa s frekvencí 30 snímků za sekundu pomoci CMOS senzoru [13]. Za příplatek je také možno sehnat NoIR variantu této kamery, která by byla vhodná pro sledování pohybu zvěře v nočních podmínkách.



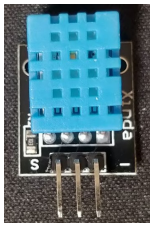

Obrázek 8: Kamera OmniVision OV5647

## 4.4 Měření teploty a vlhkosti

Z důvodu rozšíření funkcionality byl do systému zahrnut digitální senzor teploty a vlhkosti DHT11 (viz obr. 9). Senzor je schopný měřit teploty v rozmezí 0 až 50 °C s přesností měření ±2°C. U měření vlhkosti je uveden rozsah 20% - 90% RH a očekávaná přesnost ±5% RH. Senzor pracuje na napětí od 3 do 5,5 V s velmi nízkým proudem 100 - 150 $\mu A$. Komunikace s tímto senzorem musí být zahájena ze strany Raspberry Pi, v opačném případě nezašle DHT11 žádná data. Senzor zasílá odpověď o velikosti 40 bitů, přičemž část informace o teplotě a vlhkosti tvoří 16 bitů, tedy každá vyjádřená 8 bity [16]. Implementace je řešena za pomoci knihovny vytvořené pro čtení dat ze senzorů DHT11 a DHT22. Druhý zmíněný senzor je schopen měřit teplotu v rozsahu od -40 do 80 °C s přesností ±0.5°C a je vhodným kandidátem vylepšení systému s ohledem na zimní měsíce.

## 4.5 Napájení

Samotnému Raspberry Pi stačí dodat ke správné funkci napětí 5 V s proudem kolem 500 mA. U kamery se předpokládá, že bude po většinu času zapnutá. Kamera samotná vyžaduje přibližně 250 mA - využití Raspberry Pi jakožto zdroje pro některé z periferií systému proto není tou nejlepší volbou. Z toho důvodu je jako zdroj pro tyto periferie použit napájecí USB breakout



Obrázek 9: Snímač DHT11

modul s micro USB vstupem 5 V, výstupním napětím 1,8, 3,3, 5, 9, či 12 V a maximálním zatížením 500 mA.

Napájecí USB modul byl při testování po zapojení a spuštění všech komponent zatížen maximálním proudem 450 mA. Klidový proud s vypnutou službou *Motion* se pohyboval mezi 180 až 230 mA. Klientské Raspberry Pi obsluhuje pouze motor, a není tedy zapotřebí dalšího externího zdroje.

## 4.6 Zapojení systému

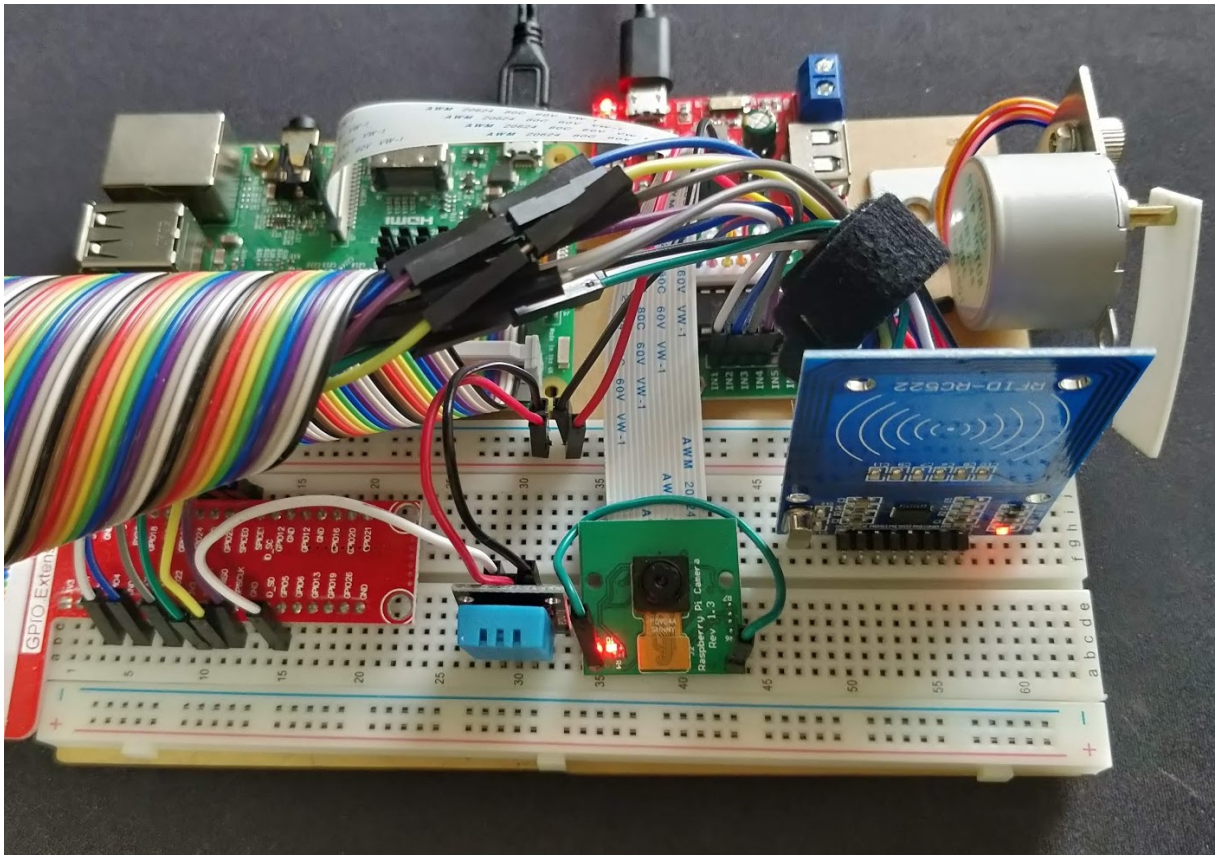
Systém je zapojen na nepájivém kontaktním poli. Pro jednodušší práci s GPIO porty byla použita deska rozšiřující GPIO piny k použití na nepájivém poli. Všechny komponenty jsou připevněny k akrylovým deskám pro lepší stabilitu (viz obr. 10).

Tabulka 1: Zapojení GPIO pinů

| Pin# | RaspberryPi | Pin# | Zařízení | Typ pinu |
|---|---|---|---|---|
| 6 | Ground | 6 | MFRC522 | Ground |
| 7 | GPIO04 | 1 | ULN2003 | IN1 |
| 11 | GPIO17 | 2 | ULN2003 | IN2 |
| 15 | GPIO22 | 3 | ULN2003 | IN3 |
| 16 | GPIO23 | 4 | ULN2003 | IN4 |
| 18 | GPIO24 | 5 | MFRC522 | IRQ |
| 19 | GPIO10 (SPI0_MOSI) | 3 | MFRC522 | MOSI |
| 21 | GPIO09 (SPI0_MISO) | 4 | MFRC522 | MISO |
| 22 | GPIO25 | 7 | MFRC522 | RST |
| 23 | GPIO11 (SPI0_SCLK) | 2 | MFRC522 | SCK |
| 24 | GPIO08 (SPI0_CE0) | 1 | MFRC522 | SDA |
| 29 | GPIO05 | 1 | DHT11 | Data |
| 39 | Ground | 3 | DHT11 | Ground |

V tabulce 1 je znázorněno přesné zapojení komponent systému do GPIO portu Raspberry Pi 3. Podle zapojení GPIO pinů modulu čtečky typu MFRC522 je vidět, že jsou připojeny piny MOSI, MISO, SS, SCK i SDA. Je tedy umožněno využití komunikačního rozhraní $I^2C$ a také SPI (viz kapitola 3.3.2). RST (reset) pin čtečky MFRC522 slouží k vypnutí komunikace na vstupu v případě je-li jeho hodnota *LOW*. Napětí výstupních pinů zůstane během vypnutí vstupních

bufferů zamrzlé na své původní hodnotě [6]. GPIO piny 7, 11, 15 a 16 jsou určeny pro driver motoru ULN2003. Tyto piny jsou nastaveny jako výstupní pro řízení stavů motoru. Dále je do portu 29 zapojen datový pin senzoru teploty a vlhkosti, který slouží k dotazování a čtení dat (viz kapitola 4.4).



Obrázek 10: Zapojení hlavního systému

# 5 Řešení implementace

Spouštění systémových funkcí jako je zapnutí čtení RFID tagů, ovládání vrat, kontrola kamery a snímače teploty je řízeno z centralizované databáze. Důvodem této implementace je odloučení funkční části aplikace od aplikace webové a tím zajištění integrity dat a jednoduché přenositelnosti systému.

Nevýhodou tohoto řešení by v praxi mohla být situace, kdy k centralizované databázi neustále přistupuje několik zařízení zároveň přes pomalé síťové připojení. Tento problém je možno částečně eliminovat zasíláním nízkonákladových zpráv s použitím protokolu MQTT. Zasíláním instrukcí z dedikovaného Raspberry Pi ostatním klientům rovněž zajišťuji distribuovanost systému bez nutnosti modifikace centrální databáze či neustálého čtení jiným kontrolním zařízením.

## 5.1 Čtení RFID tagů

Komunikovat s RFID modulem mi pomohou knihovny SPI-Py a MFRC522, která obsahuje třídu a sadu instrukcí pro čtení, zápis a autentizaci tagu pomocí rozhraní SPI. I když při použití Mifare tagu lze zapisovat až 4 kB dat, funkce zápisu a autentizace pro tento projekt vyžadovány nejsou, neboť nepotřebujeme permanentně ukládat žádné informace o objektu mimo naši databázi. Co se týče autentizace, tak žádné bezpečnostní prvky spolupracující s RFID v tomto systému navrženy nejsou. Nicméně jednoduchý návrh implementace autentizace je k vidění v kódu souboru *RFID.py*.

Nejdříve vytvoříme instanci MFRC522 a poté pomocí *Request*() metody čekáme ve smyčce na aktivaci přiloženým RFID tagem. Pokud byl tag přiložen a nedošlo k chybě, pokusíme se o získání unikátního 40bitového identifikátoru (UID) metodou *Anticoll*(). Po úspěšném načtení tagu jej uložíme pro porovnání s databází a zápisu do logu. V opačném případě se může stát, že metoda nevrátí správnou hodnotu kvůli rušení, špatné polohy tagu a nebo překážce, v takovém případě je třeba čtení opakovat.

```
# Vytvoření instance třídy pro práci s RC522
rc = MFRC522.MFRC522()
# Smyčka pro běh čtení
while loop == True:
   # Čekání na načtení RFID tagu
   (status, TagType) = rc.MFRC522_Request(rc.PICC_REQIDL)
   # Pokud je karta načtena, pokus se zjistit UID
   if status == rc.MI_OK:
      (status, uid) = rc.MFRC522_Anticoll()
   # Pokud je načteno UID, parsuj do řetězce
   if status == rc.MI_OK:
```

```
uidRead = str(uid[0])+'.'+str(uid[1])+'.'+str(uid[2])+'.'+str(uid[3])+'
    .'+str(uid[4])
```

Výpis 1: Načítání UID z tagu.

Dále se dotazujeme, zdali UID tohoto tagu v databázi již existuje. Pokud ano, zjistíme předchozí lokaci zvířete, upravíme ji společně s časem průchodu a akci vložíme do logu. Po úspěšném čtení uspíme program na půl sekundy, aby tag zvířete nebyl přečten dvakrát.

```
# Pomoci UID najdi záznam v databázi
result=db.select_object_uid(uidRead)

# Pokud bylo zvíře venku
if result[3] == '0':
   # Uprav záznam zvířete na hodnotu inside
   db.update_object_inside('1', uidRead, now)
   # Záznam o průchodu
   db.insert_log((now, uidRead, result[2], "Inside"))
   # Uspi na půl sekundy
   time.sleep(0.5)
```

Výpis 2: Porovnání UID zvířete s databází.

V případě, že tento tag ještě není v databázi, je tato aktivita rovněž zapsána do logu. Uživatel využije této funkce k identifikaci UID nových tagů při přidávání zvířat do systému.

Pro zvolený modul MFRC522 lze také nastavit maximální zisk antény registrem $RFCfgReg$ na výrobcem uváděných 48 dB (viz výpis 3). Dosah lze tímto způsobem navýšit jen o pár milimetrů.

```
>>> import lib.MFRC522
>>> rc = lib.MFRC522.MFRC522()
>>> rc.SetBitMask(rc.RFCfgReg, (0x07<<4));
>>> print rc.RFCfgReg
48
```

Výpis 3: Nastavení zisku antény MFRC522.

## 5.2 Motor

Otevírání vrátek je zajištěno motorem s převodovým poměrem 1:64. Moje implementace využívá sekvenci osmi kroků, kterou lze podle potřeby a prodlevy mezi jednotlivými stavy změnit třeba na sekvenci stavů čtyř, čímž dosáhneme dvojnásobné velikosti úhlu mikrokroku z 0,703125° na 1,40625°.

```
# Nastavení sekvence kroku
Step = {}
Step[0] = [1,0,0,0]
Step[1] = [1,1,0,0]
Step[2] = [0,1,0,0]
Step[3] = [0,1,1,0]
Step[4] = [0,0,1,0]
Step[5] = [0,0,1,1]
Step[6] = [0,0,0,1]
Step[7] = [1,0,0,1]
```

Výpis 4: Nastavení sekvence kroků motoru.

V dalším kroku je, v závislosti na zvoleném číslování pinů, nutné připravit zapojené GPIO piny k použití, a to výběrem BOARD, nebo BCM metody. Následně jsou pomocí GPIO *setup*() metody zvoleny tyto piny jako výstupní kanály. Metoda *moveCoil*() obstarává změny hodnot na výstupu GPIO v závislosti na procházené sekvenci.

```
# Definování GPIO pinu
coils = [7, 11, 15, 16] # 4, 17, 22, 23
# Příprava GPIO
GPIO.setmode(GPIO.BOARD)
# Nastavení GPIO pinů na vystup
for coil in coils:
   GPIO.setup(coil, GPIO.OUT)
# Funkce pro přepínání GPIO pinů podle sekvence
def moveCoil(Step):
   for i in range(len(coils)):
      # Nastavení výstupu pinu
      GPIO.output(coils[i], Step[i])
```

Výpis 5: Sestavení a procházení sekvencí GPIO.

Samotná metoda *closeGate*() slouží k procházení sekvencí pro nastavený počet kroků. V tomto případě je nastaveno 256 kroků, což znamená posunutí o 180°. Mezi jednotlivými sekvencemi je nastavena prodleva 2 ms. Při hodnotách prodlevy menších než 2 ms nebyl motor schopen na instrukce reagovat. Metodu *openGate*() realizujeme obdobně s tím rozdílem, že zvolíme reverzní průchod sekvencemi. Po otevření/zavření vrat je vhodné vynulovat stále aktivní GPIO piny metodou $moveCoil(0, 0, 0, 0)$.

```
# Funkce zavření vrat
def closeGate(db):
```

```
# Smyčka pro zavírání vrat s definovanym počtem kroku
for x in range (256):
   for i in range(len(Step)):
      moveCoil(Step[i])
      # Pauza 2 ms mezi sekvencemi
      time.sleep(0.002)
```

Výpis 6: Funkce zavření vrat.

## 5.3 Kamera

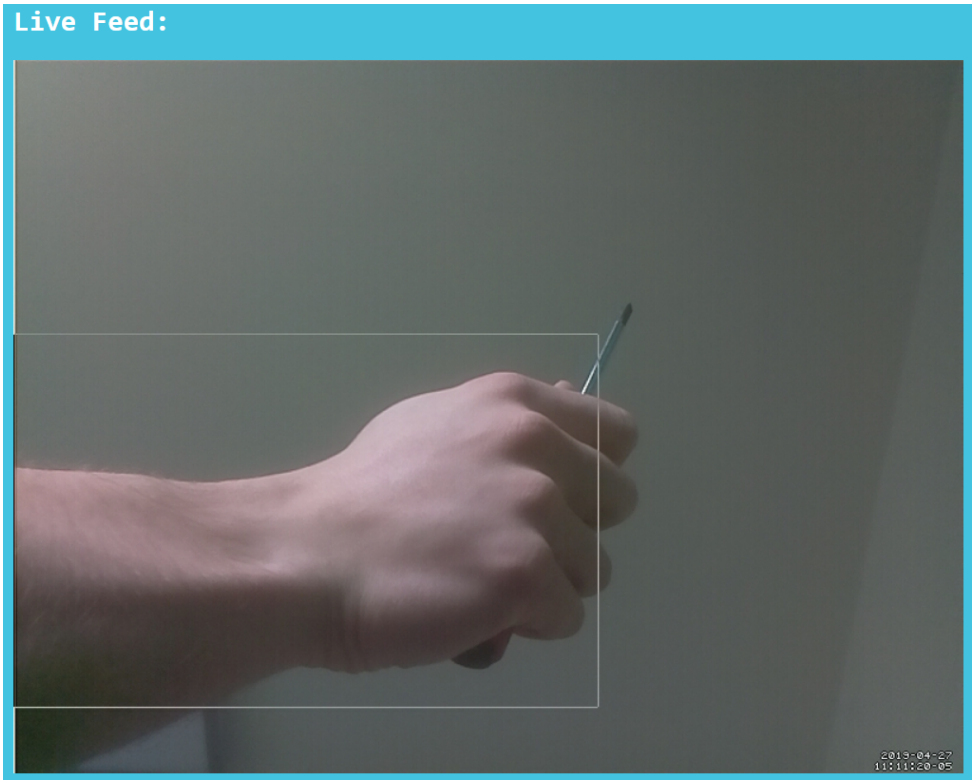
Kameře je nutné definovat příkazem *modprobe* přístupový ovladač a přidat tento ovladač do souboru */etc/modules* pro načtení použití při dalším spuštění systému. Nastavení služby jsou prováděna v konfiguračním souboru */etc/motion/motion.conf*. Je třeba zkontrolovat správnost cesty k zařízení, vypnout localhost stream, nastavit port pro streamování na http serveru a zapnout *motion* démona. V mém případě také nastavuji rozlišení a otočení přenosu a povoluji detekci pohybu.

```
# Nastavení konfiguračních parametrů
vim /etc/motion/motion.conf
   daemon on
   locate_motion_mode on
   rotate 180
   width 800
   height 600
   stream_auth_method 0
   stream_port 8081
   stream_localhost off
# Restart služby Motion
systemctl restart motion.service
```

Výpis 7: Nastavení Motion.

Posledním krokem je aplikace nastavení, kterou provedeme restartováním služby *Motion*. URL (Uniform Resource Locator) Adresu streamu mohu poté zvolit jako zdroj pro zobrazení na webové aplikaci. *Motion* při detekci pohybu vykreslí kolem dané oblasti čtverec (viz obr. 11).

Dále je například možné nastavit kvalitu streamu (*stream_quality*), udělat portrét jednou za x sekund (*snapshot_interval*), změnit snímkovou frekvenci (*stream_maxrate*) a nebo ji omezit na jeden snímek za sekundu pokud není detekován pohyb (*stream_motion*).




Obrázek 11: Kamera ve webové aplikaci

## 5.4 Databáze

V databázi existují čtyři tabulky - *animals*, *logs*, *operation* a *temperatures*. Pro správnou komunikaci s periferiemi bude muset kontrolní funkce přistupovat k databázi a zasílat dotazy na tabulku *operation*. Je tedy zajištěno, aby třída metod pro připojení a práci s databází měla v daný moment pouze jednu instanci.

```
# Singleton pro zajištění jediné instance třídy pro práci s databází
def __new__(self):
   if not self._dbInst:
      self._dbInst=super(Database, self).__new__(self)
      return self._dbInst
# Inicializace/předání instance třídy Database
def __init__(self):
   self.sConn = self.db_connect(self.Database)
```

Výpis 8: Singleton třídy *Database*.

Pro práci s SQLite databází je vytvořen objekt připojení, následně je inicializován kurzor a pomocí metody *execute*() se provede připravený SQL dotaz. Transakce je ukončena metodou *commit*().

```
# Vytvoření objektu Connect pro přístup k databázi
def db_connect(self, db_path):
    try:
        self.sConn = sqlite3.connect(db_path)
        return self.sConn
...
# Metoda pro vložení záznamu do logu
def insert_log(self, log):
    # SQL string
    sqlStr = "insert into logs (time, uid, name, note) values (?,?,?,?)"
    # Kurzor vykonávající transakci
    cursor = self.sConn.cursor()
    cursor.execute(sqlStr, log)
    # Potvrzení a ukončení transakce
    self.sConn.commit()
```

Výpis 9: Připojení a operace nad databází.

Třída *Database* definuje všechny potřebné metody pro práci s databází.

## 5.5 Komunikace přes MQTT

Se zasíláním MQTT zpráv vypomáhá klientská knihovna Paho MQTT pro Python. Z knihovny je nutné importovat třídu *Publish* a *client*(). Třídu *Publish* využije naše master zařízení pro zasílání zprávy o otevření vrátek všem poslouchajícím odběratelům. Vydavatel může zprostředkovateli publikovat jednu a nebo více zpráv pomocí metod *single*() a *multiple*(). Zprostředkovatel udržuje s klientem perzistentní spojení, dokud si sám nevyžádá spojení nové nebo jej ukončí metodou *disconnect*(). Stejně jako server, může i klient publikovat zprávy přes zprostředkovatele.

```
# Funkce pro publikování zprávy odběratelům
publish.single("Operation", "open_gate", hostname=192.168.1.51)
```

Výpis 10: Publikování zprávy přes MQTT.

Na straně klienta importujeme z knihovny *paho.mqtt* třídu *Client*. Konstruktoru této třídy můžeme definovat například id spojení, verzi MQTT protokolu a transportní protokol. Máme také možnost použít *clean_session*(), která ve $false$ stavu zajistí, že veškerá odebírána témata a zprávy ve frontě jsou uchovány i po odpojení klienta - v tomto systému není potřeba takové informace uchovávat. Metoda *subscription*() se spustí po úspěšném spojení se serverem a slouží k nastavení odebíraných témat. Zprávy jsou po přijetí zpracovány metodou *work_message*(), která podle obsahu přijaté zprávy provede nadefinovanou akci. V případě potřeby může být implementace doplněna o zpětnou odpověď publikovanou klientem.

```
# Metoda volaná po získání potvrzení o spojení se serverem
def subscription(mqttc, userdata, flags, rc):
    # Přihlášení se ke zprávám tematu Operation
    client.subscribe("Operation")
# Metoda volaná po přijetí zprávy
def work_message(mqttc, userdata, msg):
    if msg.payload == "gopen":
        # Otevření vrat
        openGate()
# Volání kontruktora klienta
mqttc = client.Client()
# Nastavení témat
mqttc.on_connect = subscription
# Práce se zprávou
mqttc.on_message = work_message
# Připojení k brokerovi
mqttc.connect("192.168.1.51", 1883, 60)
mqttc.loop_forever()
```

Výpis 11: Nastavení MQTT klienta.

### 5.6 Běh programu

Spouštění metod systému je realizováno skriptem *Run.py*. Skript se dotazuje na záznam databázové tabulky *operations* a v případě změny kontrolního záznamu provede potřebnou operaci či zavolá funkci importované knihovny.

Skript obsluhuje:

- Vytvoření instance tříd pro práci s databází, RFID čtečkou a teploměrem.
- Publikování MQTT zpráv učených pro ovládání dalších zařízení.
- Volání metod pro čtení RFID tagů.
- Otevírání a zavírání vrat - Volání metod ovládání motoru včetně kontroly plánovaného otevření/zavření.
- Kontrola služby *Motion* k ovládání kamery.
- Snímání teploty a vlhkosti - každou nově započatou minutu je volána metoda *get_temps* třídy DHT11.
- Zápis kontrolních záznamů do databáze.

Skript běží ve smyčce s časovou prodlevou 500 ms mezi jednotlivými iteracemi. Ukázkou jedné z aktivit skriptu je plánované otevření vrátek (viz výpis 12). Skript v podmínkách porovnává stavy kontrolních funkcí s hodnotami získanými z databáze. Při automatickém zavírání vrátek se porovnává rozdíl nynějšího času od nastavených hodnot pro otevírání či zavírání a v potřebnou minutu je volána metoda pro otevření motoru. Tato implementace byla zvolena kvůli možnosti zavření vrátek i po překročení zvoleného času. To může nastat pádem systému či výpadkem elektřiny.

Rovněž je provedena publikace MQTT zprávy pro otevření vrátek. Klient má na zprávu navázanou obdobnou metodu a ve velice krátké době ji provede. Záznam o provedení akce je také zaslán do databáze.

```
 # Pokud je nastaveno plánování otevírání/zavírání vrat
if ops[5] == '1':
    # Delta nyní - čas pro otevření vrat (HHMM)
   oDiff = int(now.strftime("%H%M")) - \
   int(ops[6][ops[6].find("u")+2:-1].replace(':', ''))
   # Pokud jsou vrátka zavřena a mají se tuto minutu otevřít
   if (oDiff == 0 and ops[2] == '4'):
      print "Scheduled gate opening."
      # Pošli instrukci k otevření vrat klientům
      publish_msg("Operation", "gopen")
      # Metoda pro otevření vrat
      Motor.openGate(db)
      # Vložení záznamu o akci do logu
      db.insert_log((now.strftime("%Y-%m-%d %H:%M:%S"),
      'Operation', 'Gate', "Scheduled gate opening."))
```

Výpis 12: Plánované otevření vrat.

U automatického zavírání vrátek je také možné nastavit zasílání mailu uživateli v případě, že je venku ještě nějaké zvíře (viz výpis 13).

```
...
# Pokud jsou jeste nějaká zvířata venku
if (db.select_objects_out() > 0):
   # Zaslání e-mailu uživateli
   os.system("echo 'Subject: Gate is closing with animals still outside!' |
       sendmail -v kristian.rek.st@vsb.cz")
...
```

Výpis 13: Zaslání e-mailu uživatelii.

Pro automatické spuštění *Run.py* a webové aplikace samotné jsou v inicializačním systému *systemd* vytvořeny zaváděcí jednotky služeb. Podmínkou spuštění procesů je načtení připojení k síti. Pro zajištění správné funkce má klientské Raspberry Pi nastavenou spouštěcí jednotku také. Dojde-li z nějakého důvodu k pádu některého ze systémů, budou služby spolu s MQTT spojením znovu nastartovány.

```
pi@raspberrypi:~ $ cat /etc/systemd/system/track-client.service
[Unit]
Description=Startup unit
After=network-online.target
[Service]
ExecStart=/usr/bin/python /home/pi/Run.py
Restart=on-failure
RestartSec=5s
User=pi
[Install]
WantedBy=multi-user.target
```

Výpis 14: Systemd jednotka Run.py.

Uživatel má také možnost sledovat veškerou aktivitu na výstupu konzole.

```
pi@raspberrypi:~ $ python BP/Scripts/Run.py
### Program Start ###
Reading temps
Date - 2019-04-29 22:08:03
Temp - 18
Humidity - 66

Tag detected.
UID of the tag: 192.163.124.122.101
Tag with this ID is not in the database!.

Tag detected.
UID of the tag: 18.90.237.28.185
Exists in database - ['34', "u'18.90.237.28.185'", "u'Max'", '0', "u
    '2019-04-2808:18:25'", '1', "u'2019-04-2920:17:47'"]
Animal went inside.

Opening the Gate.
```

Výpis 15: Výstup konzole Run.py.

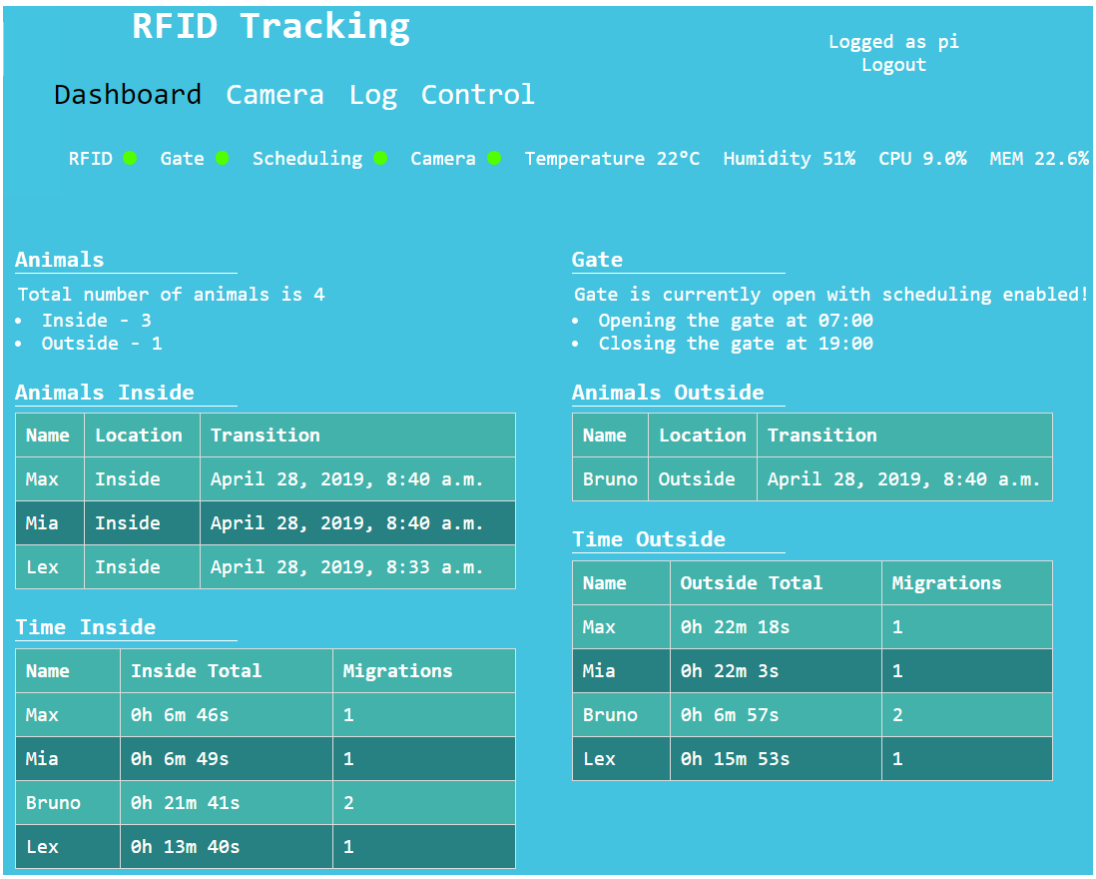
# 6 Webová aplikace

Hlavním důvodem proč jsem si vybral Django framework pro tvorbu webové aplikace je možnost využití (Model-View-Template) architektury. Každý *model* odkazuje na jednu tabulku databáze a obsahuje atributy a metody pro práci s těmito daty. Tato implementace objektově relačního mapování (ORM) je obohacena o automaticky generované API (Application programming interface), které obsahuje řadu nativních metod pro přístup a práci s daty relačních databází [17]. Značná část těchto metod mi pomohla při implementaci systému.

Django využívá svůj vlastní systém pro zobrazování dat a práci s HTML (Hypertext Markup Language) šablonami zvaný DTL (Django template language) [17]. Důležitou součástí architektury je samotný *view*, který uchovává hlavní logiku práce s daty získaných z modelu pro následné zpracování a zaslání do šablony k zobrazení.

Jak je již zmíněno u volby použitých technologií, mým cílem bylo separovat logiku webové aplikace a vyhnout se přímému volání subprocesů pro práci s periferiemi. Metody webové aplikace tedy pracují, až na pár specifických výjimek, pouze s informacemi z databáze.

Webová stránka je rozložena na čtyři hlavní pohledy - *Dashboard*, *Camera*, *Log* a *Control*. Sekce *Control* je viditelná a přístupná pouze pro přihlášené uživatele.




Obrázek 12: Webová aplikace

## 6.1 Kontrola systému

Hlavička stránek je věnována navigaci mezi jednotlivými pohledy a přihlášení uživatele. Patří k ní také část kontrolní, která signalizuje aktuální nastavení funkce periferií systému, vytíženost paměti a procesoru a také zobrazuje hodnoty teploty a vlhkosti. Před renderováním každého z pohledů je vyvolána instance třídy *Stats*, která má v konstruktoru zajištěny aktuální informace pro kontrolní sekci hlavičky.

```
class Stats():
    ...
    # Konstruktor s aktuálními atributy
    def __init__(self):
        # Poslední záznam teploty a vlhkosti
        self.lhum = Temps.objects.latest('time').humidity
        self.ltemp = Temps.objects.latest('time').temperature
        # Záznam o nastavení periferií
        self.operation = Operation.objects.get(id='1')
        # Aktuální cas
        self.now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        # Využití paměti a procesoru
        self.mem = dict(psutil.virtual_memory()._asdict())
        self.mem['cpu'] = psutil.cpu_percent()
```

Výpis 16: Kontruktor pro třídu Stats.

## 6.2 Seznam a statistiky zvířat

Hlavní pohled patří stránce *Dashboard*, která obsahuje informace o počtu registrovaných zvířat a jejich aktuální statistiky. Uživatel jedním pohledem vidí kolik zvířat je vevnitř a kolik venku, jak dlouhou dobu kde strávilo a má přehled o posledním průchodu každého zvířete. Užitečná je pro uživatele i informace o tom, zdali a na jakou hodinu bylo naplánováno zavření vrátek. Samotná stránka je každých pět sekund automaticky aktualizována.

```
# Pohled pro index stránku
def home(request):
    # Instance třídy stats
    s = stats()
    # Dotaz na všechna zvířata seřazená podle posledního času průchodu
    animals = Animals.objects.all().order_by('-time')
    # Metoda vracející list statistik zvířat
    aStats = animal_stats(request, animals)
    # Celkový počet zvířat
```

```
    t_count = animals.count()
    # Počet zvířat vevnitř
    i_count = animals.filter(inside=True).count()
    # Počet zvířat venku
    o_count = t_count - i_count
    # Render index stránky s předanými parametry statistik o zvířatech a
        informací do hlavičky stránky
    return render(request, 'tracking.html', {'s' : s, 't_count' : t_count, '
        animals' : animals, 'title' : 'Tracking', 'i_count' : i_count, '
        o_count' : o_count, 'iaStats' : aStats})
```

Výpis 17: Implementace home view.

### 6.3 Ovládací panel

Po přihlášení do systému má uživatel možnost měnit nastavení periferií. Kromě přepínání funkce čtečky, kamery a snímače teploty může uživatel také ovládat a nastavovat čas pro automatické otevírání a zavírání vrátek. Tento pohled může správce také používat pro jednoduché přidávání zvířat.

V sekci *Tools* je rovněž odkaz na přístup do administrátorského centra, kde má správce možnost přidávat další uživatele, modifikovat jejich práva, tvořit skupiny a měnit veškeré záznamy relací databáze. Dále je implementována možnost smazání všech záznamů tabulek či jen některé z nich. Tyto metody naleznou využití například při novém nasazení systému.

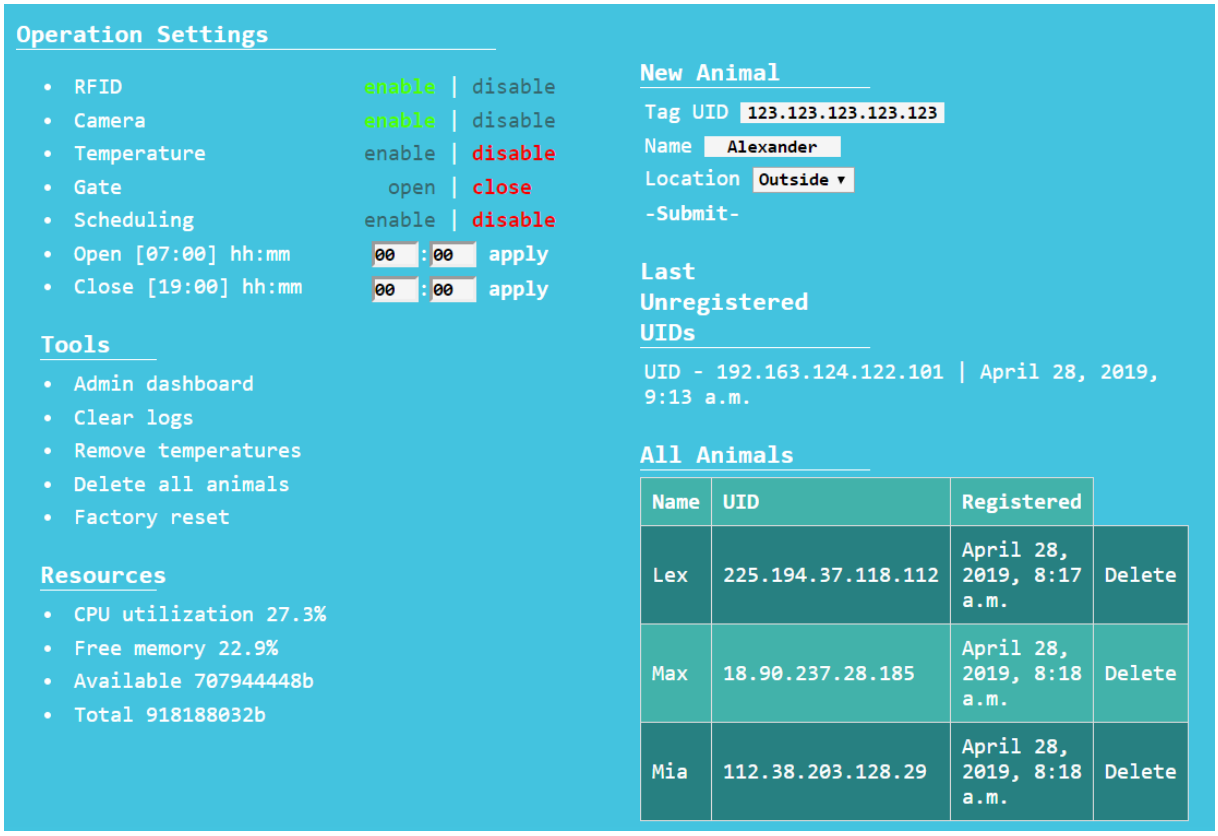
Metody implementované pro ovládací panel:

- **control()** - metoda pro vytvoření stránky *Control*. Hledá v logu záznamů posledních 5 identifikátorů tagů, které byly detekované čtečkou, ale ještě nejsou v databázi. Tato metoda slouží uživateli k ulehčení procesu přidávání nových zvířat. Identifikátory posledních neznámých tagů se vypíšou pod formulář vložení zvířete.

- **set_control()** - tato metoda slouží pro ovládání hlavních periferií systému - RFID čtečky, kamery, teploměru a vrátek. Dostává dva parametry z URL cesty, první uvádí zařízení a druhý hodnotu pro nastavení.

- **set_ctime(), set_otime()** - metody určené k nastavení času pro automatické otevření/-zavření vrátek. Reaguje na metodu *POST* formuláře. Po ověření bezpečnostního CSRF (Cross Site Request Forgery) klíče a validaci dat nastaví čas na hodnotu požadovanou uživatelem.

- **factory_reset()** - dojde k smazání všech dat v databázi kromě záznamů tabulky *operation*. Uživatel bude vyzván k potvrzení své volby.




Obrázek 13: Kontrolní panel systému

- **entry_delete()** - metoda pro mazání záznamů logu, zvířat a teplot z databáze. Argument získaný z URL cesty udává které záznamy se mají smazat.
- **animal_delete()** - smazání jednoho zvířete a všech záznamů o něm v logu. ID zvířete pro smazání je převzato z URL cesty.
- **create_animal()** - metoda vytvoření zvířete. Po vyplnění UID, jména a pozice zvíře z předdefinovaného formuláře jsou data opět validovány. Je-li zvíře již evidováno, dojde k přesměrování uživatele na výpis chybné hlášky metodou *error_message*().
- **error_message()** - přesměruje uživatele na stránku s chybovou hláškou.

Výpis 18 reprezentuje ukázku nastavení parametrů pomocí URL cesty pro ovládání komponent systému. Citlivé metody jsou zabezpečeny proti nepřihlášeným uživatelům použitím dekorátoru *login_required*(). Pokud se takový uživatel pokusí o zavolání metody, bude přesměrován na přihlašovací stránku.

```
# Zabezpečení pohledu proti zneužití URL
@login_required(login_url="/login/")
# Metoda set_control() se dvěma argumenty
```

```
def set_control(request, val1, val2):
    s = stats()
    # Upravení kontrolní hodnoty pro RFID čtení
    if val1=='0' and (val2 == '0' or val2 == '1'):
        s.operation.rfid=val2
        # Ukončení transakce
        s.operation.save()
    if val2 == '1':
        msg = "RFID Turned on."
    else:
        msg = "RFID Turned off."
    # Zápis zprávy do logu
    add_log('Operation', 'RFID', msg)
```

Výpis 18: Ovládání RFID z webové aplikace.

## 6.4 Přidružené funkce

Monitorování využití zdrojů procesoru a paměti (viz obr. 13) je provedeno knihovnou *psutil*. Knihovna navíc obsahuje metody pro kontrolu a testování disků a zobrazení široké škály informací o síťových rozhraních systému.

View *logs* obsahuje výpis záznamů logů databáze, logy se zpracovávají podle druhu zprávy a dělí se na $Info$, *Operation*, a $Unknown$. $Info$ označuje klasické operace jako jsou průchody, vytvoření a smazání zvířat uživatelem nebo otevírání vrátek. Záznamy funkcí otevření a zavření vrátek jsou zaslány pouze po vykonání dané metody. $Unknown$ logy se v systému objeví pokud čtečka narazí na neznámý tag a *Operation* záznam značí změnu operace nějaké součástí systému.

Logging

| Time | Operation | Severity | Description |
|---|---|---|---|
| April 28, 2019, 2:52 a.m. | Animal | Info | Animal Kikin with UID - 112.38.203.128.29 added. |
| April 28, 2019, 2:52 a.m. | Animal | Info | Animal Sissi with UID - 18.90.237.28.185 added. |
| April 28, 2019, 2:52 a.m. | Animal | Info | Animal Max with UID - 192.163.124.122.101 added. |
| April 28, 2019, 2:50 a.m. | 192.163.124.122.101 | Unknown | Tag with this ID is not in the database! |
| April 28, 2019, 2:50 a.m. | 18.90.237.28.185 | Unknown | Tag with this ID is not in the database! |

Obrázek 14: Log View

# 7 Návrhy na vylepšení

V této kapitole se věnuji možnostem využití dalšího hardwaru a metod implementace pro rozšíření a vylepšení kvality systému. Zmíněná vylepšení systému jsou vhodná spíše pro specifické potřeby využití a jejich zahrnutí do této práce nebylo pro funkci systému přímo vyžadováno.

## 7.1 Hardware

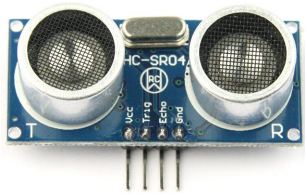
Sledování větších zvířat bude pravděpodobně vyžadovat detekci tagu na delší vzdálenosti. Použitý modul MFRC522 má velmi omezené možnosti, co se týče práce s anténou. V takovém případě by tedy bylo nutné investovat do výkonnějšího modulu. V dnešní době je k dispozici široké spektrum moderních UHF RFID systémů, které jsou schopny detekovat objekt na vzdálenost více než 15 m. Řada těchto modulů je postavena na ARM architektuře s odlehčenou verzí distribuce operačního systému Linux.

Jak jsem již zmínil v sekci věnované použitým periferiím (kapitola 4), systém je rovněž možno obohatit o funkci nočního vidění za využití kamery s NoIR filtry a přesnější teploměr DHT22 se širším rozsahem měřených teplot a vlhkosti. Toto řešení je možné například doplnit i o LCD display zapojený do DSI rozhraní pro přímé zobrazení teplot a dalších užitečných informací.

Raspberry Pi 3 model B+ může být také napájen přes Ethernetový port. Pro využití této funkcionality je však třeba zakoupit takzvaný PoE HAT a vlastnit zdroj energie kompatibilní s 802.3af standardem. Oficiální Raspberry Pi PoE HAT rovněž obsahuje větráček, který je automaticky regulovaný přes I2C rozhraní [8].

Ovládání vrat lze rovněž obohatit o senzory pro detekci překážky. Vhodnými kandidáty pro tuto funkci mohou být například:

- Rodina ultrazvukových senzorů HC-SR04, HY-SFR05, US-015 a jim podobné - obvykle detekují překážku od 1 cm do téměř 5 m.




Obrázek 15: Ultrazvukový senzor HC-SR04 v2 [18].

- Senzory využívající Hallova jevu pro detekci magnetického pole.
- Laserové senzory měřící prodlevu mezi zasláním a odrazem paprsku od objektu.
- Infračervené snímače indikující objekt po detekci zpětného IR záření (viz kapitola 4.2).


Obrázek 16: Laserový senzor překážek STM32 [19].

## 7.2 Úpravy implementace

Pro rozšíření hlavního systému o několik separátních motorů je možné jednoduchou implementací MQTT zpráv nastavit odebírání zpráv na *localhost* (kapitola 5.5). V takovém případě funguje toto zařízení zároveň jako klient. Využitím takového řešení ušetříme čas například při čekání na zavření/otevření vrátek a přepínání služby *Motion*.

Dalším možným řešením tohoto problému může být použití vláken (*threading*), *multiprocessingu* a nebo volání paralelního procesu pomocí modulu *subprocess*. Při takovémto návrhu je potřeba brát ohled na přístup ke sdíleným informacím, paměti a databázi.

Je-li nutné ovládat další funkcionality, jako je například otevírání jiných vrátek v separátní dobu a přepínání kamer, stačí upravit kontrolní záznam v databázi a doimplementovat spouštění metod/publikování MQTT zpráv ve skriptu *Run.py*.

# 8 Závěr

Cílem mé práce bylo seznámit se s možnostmi sledování objektu pomocí RFID systémů a pokusit se jeden takový systém vytvořit. Počáteční kapitoly byly zaměřeny na seznámení se s RFID technologiemi a výběrem vhodného RFID modulu pro realizaci systému. Popsal jsem výhody práce s platformou Raspberry Pi, možnosti využití GPIO portu a sériových rozhraní ke komunikaci s periferiemi. Právě už při výběru periferií nastal čas přemýšlet o volbě programovacího jazyka pro co nejčistší řešení. Všechny ukazatele mířily na Python - díky popularitě tohoto programovacího jazyka byla většina potřebných knihoven již přepsána nebo dokonce vyvinuta originálně v Pythonu. Další přirozenou volbou byl populární framework pro vývoj webových aplikací Django.

Jelikož základem práce bylo navrhnout řešení implementace vzorové, preferoval jsem dostupnější periferie s možností jednoduchého testování. Během sestavení a implementace systému jsem nenarazil na žádné vážnější problémy. Byla implementována funkcionalita čtení RFID tagů, při které jsem si zároveň vyzkoušel autentizaci a zápis na RFID tagy dvou různých typů. Dalším krokem bylo vytvoření systému pro ovládání vrátek za pomocí krokového motoru a následně také přidání další funkcionality ve formě teploměru. Vše je společně s kamerou implementováno do jedné webové aplikace, ve které má uživatel tyto funkcionality možnost kontrolovat a zároveň má kompletní přehled o pohybu sledovaného objektu či zvířete. Součástí řešení je také klientský počítač Raspberry Pi, který provádí instrukce na základě MQTT zpráv. V závěru práce jsem rovněž uvedl několik možností jak vylepšit a rozšířit tento systém.

Díky veřejně přístupné dokumentaci byla práce s jednotlivými komponenty dobře uchopitelná, pomohla mi prohloubit znalosti a také vyvolat zájem o další průzkum možností praktického využití automatizace s pomoci nově nabytých vědomostí o dostupných technologiích.

# Literatura


[1] Wikipedia: *Automatic identification and data capture* [online]. 2005. [cit. 2019-04-29]. Dostupné z: https://en.wikipedia.org/wiki/Automatic_identification_and_data_capture

[2] IMPINJ. *TYPES OF RFID SYSTEMS* [online]. 2018. [cit. 2019-03-28]. Dostupné z: https://www.impinj.com/about-rfid/types-of-rfid-systems/

[3] FINKENZELLER, Klaus. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*. Third Edition. 2010. 462 s. ISBN 10: 0470695064

[4] RFID4U. *RFID Basics & Resources* [online]. 2015. [cit. 2019-03-28]. Dostupné z: https://rfid4u.com/rfid-basics-resources/

[5] RAIN. *RAIN Technology* [online]. 2017. [cit. 2019-03-29]. Dostupné z: https://rainrfid.org/technology/

[6] NXP Semiconductors N.V. *MFRC522 Standard performance MIFARE and NTAG frontend* [online]. 2016. [cit. 2019-04-01]. Dostupné z: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf

[7] UPTON, Eben a HALFACREE, Gareth. *Raspberry Pi® User Guide*. 2nd Edition. 2014. 314 s. ISBN 978-1-118-79548-4.

[8] RASPBERRY PI FOUNDATION. *Raspberry Pi hardware* [online]. 2019. [cit. 2019-04-02]. Dostupné z: https://www.raspberrypi.org/documentation/

[9] CIRCUIT BASICS. *BASICS OF UART COMMUNICATION* [online]. 2016. [cit. 2019-04-02]. Dostupné z: http://www.circuitbasics.com/basics-uart-communication/

[10] DHSERVIS. *Pulsně šířková modulace* [online]. 2003. [cit. 2019-04-04]. Dostupné z: http://www.dhservis.cz/psm.htm

[11] GM Electronic. *Krokový motor + řídící modul* [online]. 2003. [cit. 2019-04-07]. Dostupné z: https://www.gme.cz/krokovy-motor-ridici-modul

[12] New Japan Radio Co.,Ltd. *Microstepping* [online]. 2016. [cit. 2019-04-07]. Dostupné z: https://www.njr.com/semicon/PDF/application_notes/Microstepping_APP_E.pdf

[13] AUTY, Ian. *OmniVision OV5647 Camera Module* [online]. 2018. [cit. 2019-04-16]. Dostupné z: https://github.com/techyian/MMALSharp/wiki/OmniVision-OV5647-Camera-Module

[14] MOTION Project. *About Motion* [online]. 2016. [cit. 2019-04-16]. Dostupné z: https://motion-project.github.io/

[15] ARITS, Martin. *MQTT Comment Message* [online]. 2016. [cit. 2019-04-19]. Dostupné z: https://lists.oasis-open.org/archives/mqtt-comment/201608/msg00002.html

[16] GAY, Warren W. *Mastering the Raspberry Pi*. 1st Edition. 2014. xxxi. ISBN-10: 1484201825.

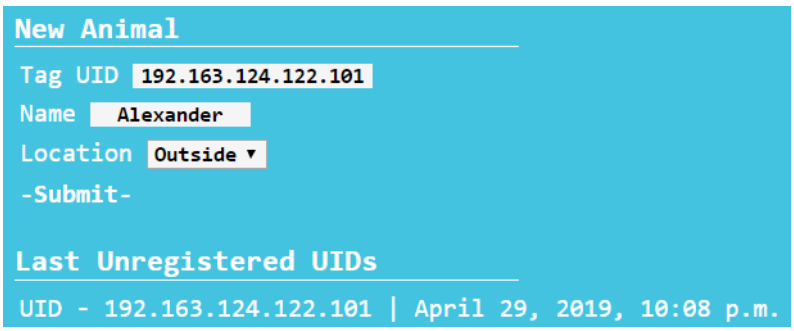
[17] Wikipedia: *Serial Peripheral Interface* [online]. 2018. [cit. 2019-04-20]. Dostupné z: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface

[18] SGBOTIC. *HC-SR04 v2* [online]. 2018. [cit. 2019-04-22]. Dostupné z: https://www.sgbotic.com/index.php?dispatch=products.view&product_id=1855

[19] LASKARDUINO. *ARDUINO LASEROVÝ SENZOR PŘEKÁŽEK* [online]. 2016. [cit. 2019-04-22]. Dostupné z: https://laskarduino.cz/vstupni-periferie-cidla/131006-laserovy-senzor-prekazek.html

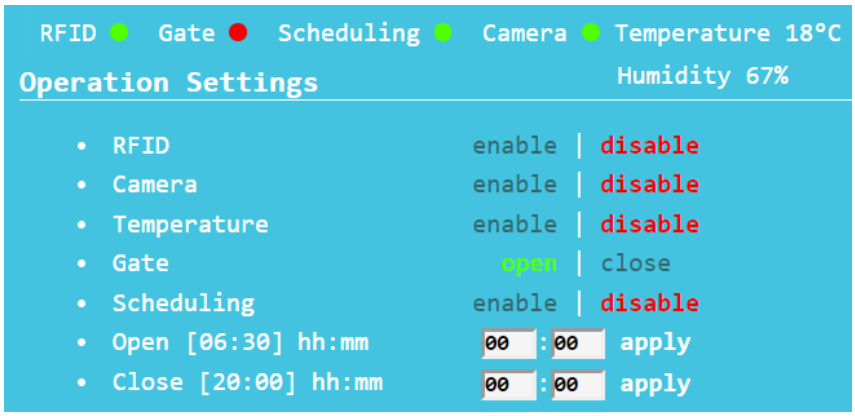
# A Možný scénář použití

Uživateli se líbí představa vlastních bio vajíček. Pořídil si proto malý zahradní kurník a čtyři slepice. A jelikož chce mít vše pod kontrolou, sestavil si tento RFID systém. Slepicím chtěl na nohy připevnit tagy ve formě plastového štítku. Jelikož na štítku nebylo nic napsáno, přiložil tento tag k RFID čtečce, přihlásil se do webové aplikace a podle záznamu času identifikace nezaregistrované karty zjistil UID tagu. Toto UID zadal společně s novým jménem a výchozí pozicí slepice do systému (viz obr. 17). Jména slepicím pro jistotu napsal na štítky.




Obrázek 17: Scénář - vložení slepice.

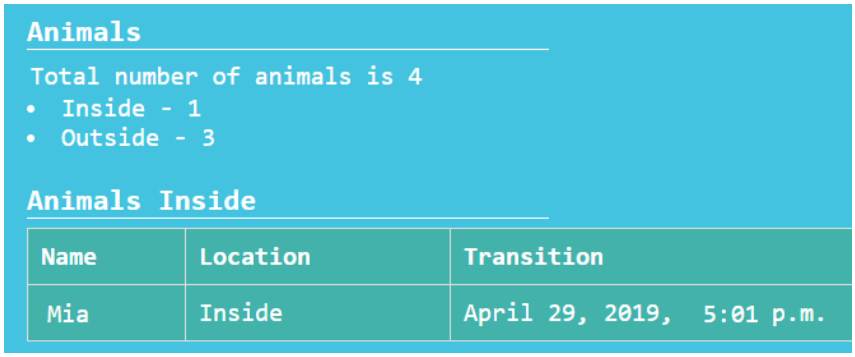
Jelikož se slepice nejpozději v 19:00 vždy vrátí samy do kurníku, uživatel rovněž nastavil automatické zavírání vrátek na 20:00. Čas pro otevření zvolil na 6:30 ráno. Uživatel je velmi spokojený se systémem a o slepice se, kromě občasného dokrmování, takřka nemusí starat.




Obrázek 18: Scénář - plánování otevírání vrátek.

Jednoho dne si ovšem při pravidelné kontrole statistik na webové aplikaci uživatel všimne, že čtečka zachytila jednu ze slepic naposledy více než 24 hodin zpátky a od té doby nedetekovala žádný její pohyb (viz obr. 19). Po kontrole kurníku vidí slepici již mrtvou.

Uživatel si vzpomene, že v konfiguraci služby *Motion* nastavil možnost *output_pictures* pro ukládání snímků při detekci pohybu. Po shlédnutí snímku vidí kunu jak vnikla do kurníku už v 19:45 předešlého dne a jeho slepici zakousla.




Obrázek 19: Scénář - seznam zvířat v kurníku.

Incident nahlásí myslivcům a doufá, že ti situaci vyřeší. Mezitím naplánuje pro jistotu zavírání vrátek už na 19:00. Odstraní tag z uhynulé slepice, smaže ji z databáze a pořídí si novou.

# B Seznam komponent a vybavení

Tabulka 2 uvádí seznam a orientační pořizovací ceny vybavení. Seznam zahrnuje i klientský Raspberry Pi a dva další typy testovaných modulů. Všechny ceny jsou vzaty z českých e-shopů. Ceny propojovacích vodičů a drobného pomocného materiálu nejsou uvedeny.

Tabulka 2: Seznam a přibližná cena vybavení

| Použité vybavení | Cena (Kč) |
|---|---|
| Raspberry Pi 3 Model B 64-bit 1GB RAM | 863 |
| Raspberry Pi 3 Model B+ 64-bit 1GB RAM | 899 |
| RFID Modul MFRC522 s kartou a čipem | 85 |
| 2x Motor 28BYJ-48 + driver modul ULN2003 | 262 |
| Kamera OV5647 pro Raspberry Pi v1.3 | 355 |
| Digitální teploměr DHT11 (3-pinový) | 69 |
| Nepájivé pole (830 pinů) | 59 |
| Cena použitého vybavení | 2 592 |
| | |
| Testované/volitelné vybavení | Cena (Kč) |
| RFID modul RDM6300 | 139 |
| 2x IR senzor vzdálenosti s detekcí překážek | 62 |
| Rozšiřující Raspberry GPIO modul | 183 |
| 2x Akrylová deska pro Arduino | 128 |